
From TwoMax to the Ising Model: Easy and Hard Symmetrical Problems

Clarissa Van Hoyweghen
Intelligent Systems Lab
University of Antwerp
Groenenborgerlaan 171
2020 Antwerp, Belgium
hoyweghe@ruca.ua.ac.be

David E. Goldberg
Illinois Genetic Algorithms Laboratory
University of Illinois
117 Transportation Building
104 S. Mathews Av. Urbana, IL 61801
deg@illgal.ge.uiuc.edu

Bart Naudts
Intelligent Systems Lab
University of Antwerp
Groenenborgerlaan 171
2020 Antwerp, Belgium
bnaudts@ruca.ua.ac.be

Abstract

The paper shows that there is a key dividing line between two types of symmetrical problems: problems for which a genetic algorithm (GA) benefits from the fact that genetic drift chooses between equally good partial solutions, and problems for which all equally good partial solutions have to be preserved to find an optimum. By analyzing in detail the search process of a selectorecombinative GA optimizing a TwoMax and comparing this search process with that of a one-dimensional Ising model, the paper investigates the difference between these two types of symmetrical problems. For the first type of problems, naively adding a niching technique to the genetic algorithm makes the problem harder to solve. For the last type of problems, niching is necessary to find an optimum.

1 INTRODUCTION

In the context of optimization by genetic algorithms (GAs), scaling, deception, epistasis, and noise are well known examples of problem difficulty characteristics. The presence of one such characteristic in the representation of a search problem indicates a certain type of difficulty the GA is to encounter during its search for global optima. In this paper we investigate another aspect of problem difficulty: the existence of equally good partial solutions in symmetrical or hierarchical problems. The loss of some equally good partial solutions due to genetic drift can prevent a GA to find an optimum, but for some problems it can also be beneficial.

The purpose of this paper is twofold. Firstly, the paper analyzes the search process of a GA solving a multimodal equivalent of the OneMax problem, the so called TwoMax, and compares its search with the search process of a GA

solving a OneMax. Secondly, the paper shows that there is a key dividing line between two different types of symmetrical problems: problems for which the GA benefits from the fact that genetic drift chooses between equally good partial solutions, and problems for which all equally good partial solutions have to be preserved to find an optimum and for which niching becomes a necessity. Naively adding a niching technique to an algorithm optimizing a problem of the first type makes the problem harder to solve.

The paper is structured as follows. Section 2 introduces equally good partial solutions or non-inferior BBs and shows how genetic drift chooses between them. Section 3 analyzes the search process of an easy symmetrical problem, a TwoMax, and shows that there exists a class of problems that benefits from the fact that genetic drift chooses between equally good partial solutions. Section 4 and 5 compare two types of symmetrical problems: problems for which searching in terms of non-inferior BBs is necessary and problems that benefit from the fact that genetic drift chooses between non-inferior BBs. Section 6 summarizes and concludes the paper.

2 NON-INFERIOR BBS AND NICHING

The working of a genetic algorithm can be explained by the search for superior building blocks. Building blocks (BBs) with above average fitness are combined to construct higher order building blocks. However, recent studies [16, 12, 7, 14] show that the search for superior BBs is not always sufficient to find a solution. When an optimization problem contains BBs which are equally good and superior to all alternatives, so called *non-inferior BBs* [4], making a choice between such BBs can avoid a GA to reach an optimum. A search strategy preserving all non-inferior BBs can then become a necessity to solve the problem quickly, reliably, and accurately. In [14], for example, it is shown that the simple GA cannot solve the standard Ising model in a reasonable amount of time because the population loses some non-inferior BBs due to genetic

drift. When a niching technique is used, the non-inferior BBs are preserved and an optimum is reached quickly. The same phenomenon can be seen when solving the H-IFF problem [16]. H-IFF combines non-inferior BBs with a hierarchical problem structure. Without a niching technique the simple GA is unable to solve the problem up to the highest level. Another class of functions which question the algorithm’s search for superior BBs is the Tobacco Road functions [3, 4]. The Tobacco road functions combine hierarchy, multimodality, and deception. At the lower level the problem can be decomposed into several non-overlapping folded trap functions, each having two complementary optima: ‘all 0s’ and ‘all 1s’. At the higher level the problem consist of one or more trap functions with solution ‘all 1s’ and deceptive trap ‘all 0s’. Without keeping all alternative low-level partial solutions, an algorithm is not able to solve the problem.

Next section describes in more detail the effect of genetic drift on non-inferior BBs and shows how niching can prevent that genetic drift chooses between them.

2.1 NON-INFERIOR BBS AND GENETIC DRIFT

A BB is called superior if it has a higher average fitness value than its competitors. For example, if

$$\begin{aligned} f(*0*) &= 1, \\ f(*1*) &= 2, \end{aligned} \quad (1)$$

we say that BB $*1*$ is superior to BB $*0*$. Non-inferior BBs are BBs which are equally good or superior to all alternatives. For example, if

$$\begin{aligned} f(*01*) &= 1, \\ f(*10*) &= 1, \\ f(*00*) &= 2, \\ f(*11*) &= 2, \end{aligned} \quad (2)$$

BBs $*00*$ and $*11*$ are non-inferior BBs. They have equally good fitness averages and their fitness average is superior to the fitness averages of BBs $*01*$ and $*10*$. When optimizing a problem containing equally good partial solutions (or non-inferior BBs), we distinguish two types of genetic search: searching for superior BBs and searching for non-inferior BBs. In the first case, the algorithm blindly chooses between equally good partial solutions. In the second case, equally good partial solutions stay in the population with equal proportions.

Consider following examples: A population contains BBs, 00, 11, 01, and 10, each occupying one fourth of the population.¹ The fitness value of 00 and 11 equals 2, whereas the fitness value of 01 and 10 equals 1. Each generation, a

¹In the example, a BB can be seen as an individual and the term fitness average can be replaced by fitness value.

new population is created as follows. Randomly pick two BBs with replacement and add the BB with the highest fitness to the new population. When two BBs have equal fitness, one of them is randomly picked and added to the new population. This selection strategy is known as binary tournament selection with replacement. Figure 1 (a) shows the result of this selection process. Clearly, the process can be divided into two phases (separated in the figure by a vertical line). During the first phase, there are two types of selection: selection for good reason between BBs with different fitness, and selection due to genetic drift between BBs with equal fitness. During this phase, BBs 00 and 11 takeover the population. Notice that at the end of the first phase BBs 00 and 11 already occupy a different proportion of the population. During the second phase, there is only selection due to genetic drift. Both BBs have equally good fitness; hence, there is no reason to prefer one above the other. The stochastic errors made by randomly picking one of them during each equal tournament cause that one of the two non-inferior BBs disappears out of the population. This example reminds us of the fact that the standard selection process works in term of superior BBs: BBs with high fitness average are preferred above BBs with low fitness average, but the selection process cannot guarantee that non-inferior BBs stay in the population with equal proportion. A well-know technique to preserve multiple good

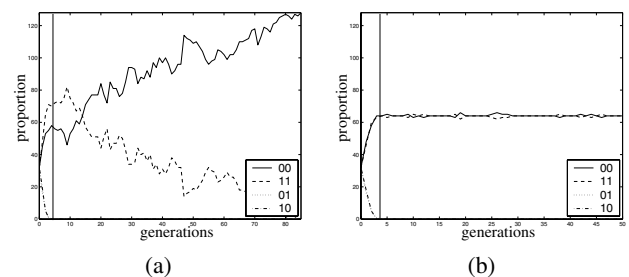


Figure 1: The plot shows the number of representatives of the BBs 00, 11, 01 and 10 during the selection process. Initially, each BB has 32 representatives in the population. In (a) no niching technique is used. In (b) each BB has to share its fitness with all BBs in the population which are not further away than Hamming distance 1.

solutions is niching. Niching tries to allocate subpopulations among non-inferior BBs such that the selection process works more in terms of non-inferior BBs instead of superior BBs. The niching technique used in this paper is fitness sharing [6], as detailed in section 5. Figure 1 (b) shows the result of the selection process using binary tournament selection with continuously updated sharing [10]. Each BB shares its fitness with all BBs in the population that are not further away than Hamming distance 1. The selection process has again two phases. During the first phase, BBs 00 and 11 takeover the population. During the second phase, the non-inferior BBs 00 and 11 are kept in

the population in equal proportion. This second example shows that when optimizing a problem using tournament selection with continuously updated sharing equally good partially solutions are selected for reproduction in the same proportion and the GA will evolve in terms of non-inferior BBs, keeping equally good partial solutions in the population.

The Ising model, H-IFF, and the Tobacco Road functions ask a search strategy that works more in terms of non-inferior BBs instead of superior BBs. Adding a niching technique to evolutionary algorithms is a possible way to get this behavior. In the context of designing a robust problem solver which solves problems of bounded difficulty quickly, reliably, and accurately, one can therefore ask the question if niching should be a standard component of a competent GA. However, in next section we will see that there exists a class of problems that benefit from the fact that genetic drift chooses between equally good partial solutions and become harder when a niching technique is naively used.

3 ANALYSIS OF A TWOMAX

The OneMax or bit-counting problem is well known and well studied in the context of GAs. In this paper, we analyze the search process of a simple GA solving a multimodal equivalent of the OneMax, the TwoMax (also called the Twin-Peaks problem [2]), and compare its search with the search process of a GA optimizing a OneMax. The OneMax problem is defined by

$$f_{OneMax}(s) = u, \quad (3)$$

with u the number of ones in the string s . The optimum is reached in the all 1s string. The TwoMax problem is defined by

$$f_{TwoMax}(s) = \left| \frac{\ell}{2} - u \right| + \frac{\ell}{2}, \quad (4)$$

with u the number of ones in the string s and ℓ the string length. A TwoMax returns the number of ones if there are more ones than zeros. If there are more zeros than ones, it returns the number of zeros, $\ell - u$. Figures 2(a) and (b) show both fitness functions with string length 100.

A TwoMax has two optima, the string containing only ones and the string containing only zeros. Moreover, a TwoMax contains *spin-flip* or *bit-flip symmetry* (see [9] in the context of GAs), a symmetry which is characterized by fitness invariant permutations on the alphabet and found in the Ising model and H-IFF. The average fitness of a schema is equal to the average fitness of the schema's complement, e.g. $f(*1*01*) = f(*0*10*)$. The problem therefore contains many equally good partial solutions or non-inferior BBs.

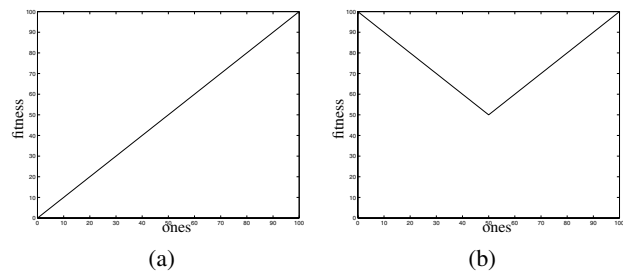


Figure 2: (a) OneMax and (b) TwoMax with string length 100

Our primary interest in this paper is to study the diversity of the population and the preservation or extinction of non-inferior BBs. The GAs used in the rest of the paper are so-called *selectorecombinative* GAs – they use only selection and recombination. To emphasize that the GA does not use mutation, we use the abbreviation srGA when confusion is possible. The GA does not use a mutation operator to add diversity to the population since mutation has only a fairly local scope and it cannot guarantee that equally good partial solutions are kept in the population. Unless stated otherwise, all experiments use binary tournament selection together with uniform crossover. The crossover rate is set to 1, and the population size equals 200.

3.1 CONVERGENCE MODEL

When solving a problem, it is useful to know how long it takes before a population converges to a population containing only optimal strings. In this section we construct a convergence model for a TwoMax using the convergence models for OneMax [8, 13]. By doing so, we gain important insight in the search process of an srGA optimizing a TwoMax.

We start by giving a quick reminder of how the convergence model for OneMax is derived. The *response to selection* equation,

$$\Delta f = f_{t+1} - f_t = b_t I \sigma_t, \quad (5)$$

introduced into the evolutionary computation community by Mühlenbein and Schlierkamp-Voosen [8], calculates the difference in mean fitness of two successive populations using the *selection intensity* I , the population's fitness variance σ_t^2 , and the *heritability* b_t . For a OneMax the heritability b_t is known to be constant and equal to 1. This means that the average fitness of the offspring is equal to the average fitness of the selected parents. The OneMax problem has an interesting characteristic: one single proportion value p_t , giving the percentage of ones in the population at generation t , reveals a lot of information about the population at generation t . If we assume that the proportion of ones is uniformly divided among the loci

in the population, the mean fitness at generation t equals ℓp_{1_t} and the fitness variance σ_t^2 can be approximated by $\ell p_{1_t}(1 - p_{1_t})$. The population is optimal when $p_{1_t} = 1$. Equation 5 now yields

$$p_{1_{t+1}} - p_{1_t} = \frac{I\sqrt{p_{1_t}(1 - p_{1_t})}}{\sqrt{\ell}}. \quad (6)$$

Approximating the above difference equation with a differential equation and integrating this equation gives

$$p_{1_t} = \frac{1}{2} \left(1 + \sin\left(\frac{It}{\sqrt{\ell}} - \theta_0\right) \right), \quad (7)$$

with $\theta_0 = \arcsin(1 - 2p_{1_{t=0}})$. The convergence time ($p_{1_t} = 1$) can then be calculated by

$$t_{conv} = \left(\frac{\pi}{2} + \theta_0\right) \frac{\sqrt{\ell}}{I}. \quad (8)$$

The above derivation holds for all selection strategies which select individuals based upon their rank in the population. For a random initialized population, $p_{1_{t=0}} = 0.5$ is a reasonable approximation and gives $\theta_0 = 0$. Using binary tournament selection, the selection intensity approximates 0.5642 [13] and the expected convergence time for a 100 bit OneMax becomes 27.84 generations.

Figure 3 plots the model for a 100 bit OneMax and compares this with experimental results. The experiments converge a bit slower due to the build-up of covariances between the alleles. The model becomes more accurate when the alleles are decorrelated by recombining twice at each generation. More details about the creation of genetic covariance together with a refined model for OneMax can be found in [13]. For this paper there is no need to use a more detailed model than the one of Eq. 7.

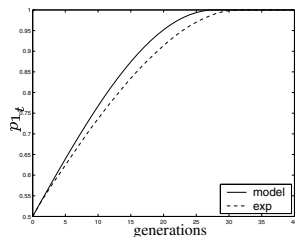


Figure 3: Convergence model and experimental results of the proportion of ones when optimizing a OneMax using tournament selection and uniform crossover. The results are averaged over 50 independent runs.

The convergence model of a OneMax predicts the proportion of ones in the population at every generation. During search this proportion p_{1_t} rises from 0.5, initial population, to 1, optimal population. The spin-flip symmetry in a TwoMax makes it impossible to use p_{1_t} to construct

a convergence model for a TwoMax; it has multiple values corresponding to an optimal population. In particular, $p_{1_t} = 1$ corresponds to the optimal population containing only ‘all 1s’ and $p_{1_t} = 0$ corresponds to the optimal population containing only ‘all 0s’. Therefore, we create a new proportion value p_t ,

$$p_t = \left| \frac{1}{2} - p_{1_t} \right| + \frac{1}{2}, \quad (9)$$

which gives the proportion of the value which appears the most in the population. When $p_t = 1$, the population contains only ones or only zeros and is converged to one of the optimal populations. For $p_t = 0.5$ several situations are possible including two extreme ones. Firstly, $p_t = 0.5$ corresponds to a population containing only strings with an equal amount of ones and zeros. This population has a minimal mean fitness, $\frac{\ell}{2}$. Secondly, $p_t = 0.5$ corresponds to a third optimal population containing both optima in equal proportion. However, we will see that an srGA without niching never converges to an optimal population containing both optima; therefore p_t will suffice to analyze the search process of a TwoMax.

Before using proportion value p_t to analyze the search process of an srGA optimizing a TwoMax, we apply it to the OneMax problem. Note that for a OneMax individuals with more ones have a higher fitness and are therefore favored by the selection strategy to produce more offspring. Hence, p_{1_t} will always increase during search and its value will never be smaller than its initial value 0.5. This allows us to replace p_{1_t} by p_t in equation 7, yielding convergence model

$$p_t = \frac{1}{2} \left(1 + \sin\left(\frac{It}{\sqrt{\ell}} - \theta_0\right) \right), \quad (10)$$

with $\theta_0 = \arcsin(1 - 2p_{t=0})$.

Figure 4 compares the search process of an srGA optimizing a OneMax with the search process of an srGA optimizing a TwoMax using proportion value p_t . The figure shows that except for a small delay during the first few generations in case of the TwoMax, both searches look very similar. In the next section we will analyze the differences and similarities of both searches in more detail.

3.2 TWO PHASES OF A TWOMAX

The individuals in a population running on a TwoMax can be divided into two niches: *niche 0*, containing only individuals with more zeros than ones, and *niche 1*, containing only individuals with more ones than zeros. Experiments show that when optimizing a TwoMax using a selective-combinative GA only one of the two optima is found. One niche disappears out of the population which allows us to divide the search process of a TwoMax into two phases:

a first phase during which both niches are still in the population, and a second phase with one niche only.

The first phase starts with a randomly initialized population. Both niches are present in the population, and $p_t = 0.5$. Parents of different niches are likely to produce poor offspring. Their children have an almost equal amount of ones and zeros, and their fitness value varies around $\frac{\ell}{2}$. This slows down the creation of fit offspring. The only good offspring are created by recombination of parents from the same niche. After a few generations one niche will contain more and fitter individuals than the other niche and takes over the population. Figure 5 shows an example of how one niche takes over the population. After 8 generations niche 1 has left the population. Figure 6 (a) shows for each niche the number of individuals and the number of individuals with a fitness value above average. The total number of individuals in the population equals 200.

To estimate the duration of the first phase, we experimentally determine the time when one niche dies out. The results are averaged over 500 runs. Figure 6 (b) shows that the first phase scales up like the *takeover time* [5]. This can be understood as the speed with which a selection strategy takes an initial proportion of best individuals to a substantial share in the population. For binary tournament selection, the time it takes the best individual to take over the population can be written as

$$t^* = \frac{\ln(n) + \ln(\ln(n))}{\ln(2)}, \quad (11)$$

with n the population size. For population size 200, $t^* = 10.049$ generations. Experimental verification shows that for this population size on average one niche disappears after 9.586 generations. In general the first phase takes a bit less than takeover time since the fittest individuals of a niche do not have to takeover the whole population, they only have to make sure that the other niche disappears out of the population.

It is important to note that we cannot use the convergence model with p_t to model the search process during the first phase because the necessary side conditions are not fulfilled. To derive the convergence model from the response to selection equation (equation 5), it is vital that ℓp_t approximates f_t and $\ell p_t(1 - p_t)$ approximates σ_t^2 . This is not the case in the first phase. Although, it is possible to end the first phase a bit earlier, for example, when one niche has (almost) no individuals with fitness average above the population's mean fitness.

The second phase is easier to analyze. The search process is now similar to the search process of a OneMax and we can use the convergence model described in equation 10 to model the search and to estimate the duration of

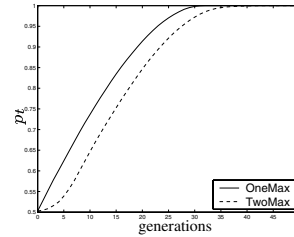


Figure 4: Experimental results of the proportion value p_t when optimizing a OneMax and a TwoMax. All results are averaged over 50 independent successful runs.

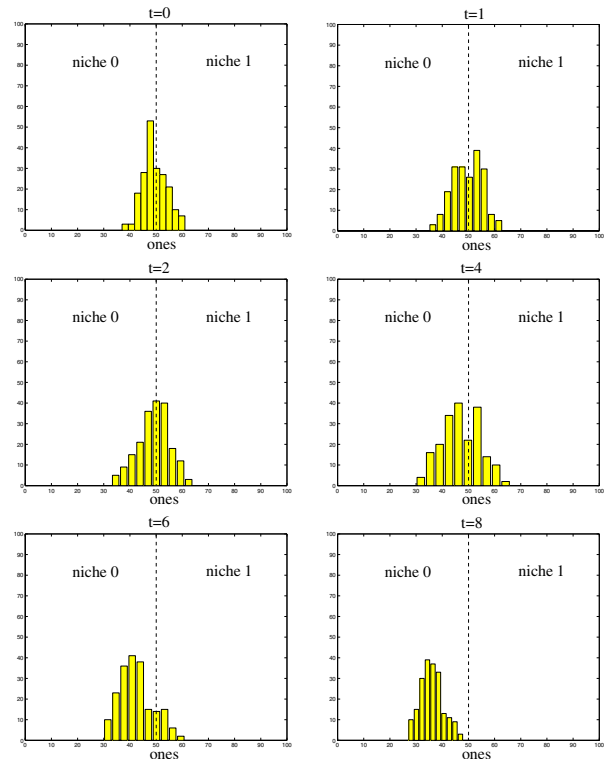


Figure 5: An srGA running on a TwoMax. The histograms have the number of ones in an individual on the horizontal axis and the number of corresponding individuals in the population on the vertical axis. Generations 0, 1, 2, 4, 6 and 8 are displayed.

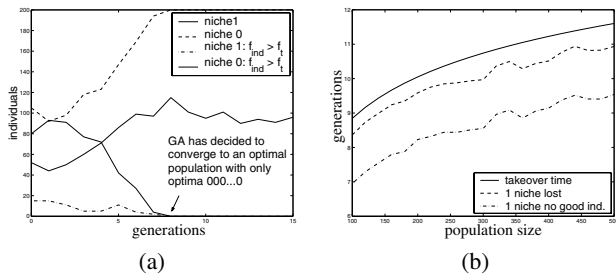


Figure 6: (a) shows, for each niche, the number of individuals and the number of individuals with a fitness value above average. (b) shows when one niche has no individuals anymore with fitness value above average and when one niche leaves the population. The results in (b) are averaged over 500 runs.

the second phase. Experiments show that for population size 200, p_t equals 0.6232 at the beginning of the second phase (after about 9 generations). Plugging $p_{t=0} = 0.6232$ ($\theta_0 = -0.2490$) into equation 10 yields the convergence model for the second phase. The calculated convergence time for the second phase is 23.4284 generations. Figure 7(a) shows the convergence model for the second phase of a 100-bit TwoMax and compares this with experimental results. The experiments again converge a bit slower due to the build-up of covariances between the alleles. Figure 7(b) shows the end of the first and the second phase for a 100-bit TwoMax using different population sizes. Note that the algorithm finds always an optimum. The results are averaged over 500 runs. The figure shows also the time when the first optima is found.

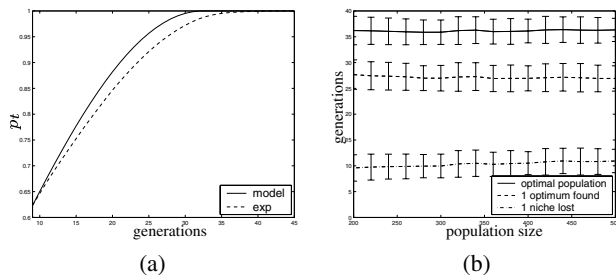


Figure 7: (a) Convergence model and experimental results of the p_t when optimizing a TwoMax using tournament selection and uniform crossover. The results are averaged over 50 independent runs. (b) The figure shows the end of the first phase, the time when the first optimum is found and the end of the second phase. The results are averaged over 500 runs.

4 DIFFICULT SYMMETRICAL PROBLEMS

Previous sections showed that when optimizing a TwoMax, the srGA chooses between equally good solutions and ben-

efits from this choice. The population converges quickly to an optimal population containing only one optimum. This is not the case for the search process of an Ising model, a search problem which can be defined by

$$f_{Ising}(s) = \sum_{i=1}^{\ell} \delta(s_i, s_{i+1}), \quad (12)$$

with string length ℓ , $s_{l+1} \equiv s_1$ and

$$\delta(s_i, s_j) = \begin{cases} 1 & \text{if } s_i = s_j, \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

The problem contains spin-flip symmetry and has as optima the string containing only ones and the string containing only zeros. The loss of non-inferior BBs in the population prevents a GA (with or without a mutation operator) to optimize an Ising model. In this section we explain the difference in difficulty between easy symmetrical problems, like a TwoMax, and a hard symmetrical problem, like the Ising model. For a more in depth analysis of the Ising model and its difficulties we refer to [14].

Symmetrical problems contain multiple non-inferior BBs. Due to the finite population size, stochastic errors cause that some non-inferior BBs disappear out of the population. This can prevent a GA to find the optimum, as it does in case of the Ising model or H-IFF. The GA gets stuck in a local optimum because the schemata in its current population, which are non-inferior BB of different optima, cannot be combined to form an optimum, and the mutation operator is not strong enough to add diversity to the population. We say that the GA has a *synchronization problem* [15]. Figure 8 shows an srGA with population size 10000 having a synchronization problem while optimizing an Ising model. The diversity measure at the z-axis represents the probability that an individual in the population has a value 1 at a certain string position. The population is not diverse enough to create an optimum.

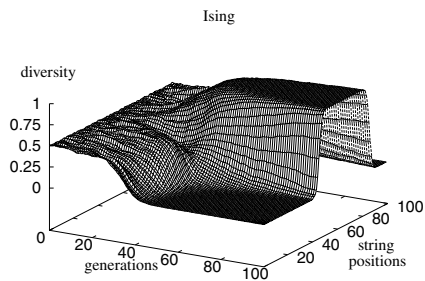


Figure 8: Diversity plot of an srGA optimizing an Ising model with population size 10000. The algorithm uses only two-point crossover.

Comparing the BBs of an Ising model with those of a TwoMax explains why GAs running on the Ising model are more likely to encounter synchronization problems than GAs running on a TwoMax. If we look at BBs whose fixed positions are not all adjacent, we note that BBs like $*00*11*$, which are not BBs of an optimum, and BBs like $*11*11*$, which certainly are BBs of an optimum, have the same average fitness value for the Ising model because there is no interaction between the non-adjacent string positions. This means that they both have the same probability to be selected for reproduction, although one of them is not a BB of an optimum and leads the algorithm to a synchronization problem when diversity is lost. This is not the case for a TwoMax. For TwoMax, BBs with equal values at all fixed string positions have a higher average fitness value than other BBs. The algorithm decides faster to which of the two optima it converges and when diversity is lost, a well sized population still contains the necessary BBs to construct an optimum.

Symmetrical problems which frequently encounter synchronization problems are simple to characterize. In contrary to easy symmetrical problems, an increase in fitness does not automatically imply a decrease in Hamming distance towards the global optimum. Typically, they have highly fit candidate solutions which are at hamming distance far from an optimum and are usually combinations of different optima. Many hierarchical problems satisfy these characteristics and encounter synchronization problems when optimized. The Ising model shows that synchronization problems can also appear when optimizing non-hierarchical problems. A useful strategy to avoid synchronization problems and to keep all non-inferior BBs in the population is niching.

5 NICHING

Fitness sharing [6] is a well-know niching technique that accomplishes subpopulations or niches by degrading an individual's fitness according to the similarity with other individuals in the population. Section 2.1 showed that when using binary tournament selection with continuously updated sharing, the selection process works more in terms of non-inferior BBs and equally good solutions are kept in the population. In this section we try to find the two optima of respectively a 100-bit TwoMax, and a 100-bit Ising model using an srGA with this sharing method. To avoid linkage problems on the Ising model, the algorithm uses two-points crossover. As sharing function the *triangular sharing function* defined in [1] is used. The sharing threshold is calculated by $\sigma_{sh} = \frac{1}{2}(\ell + \frac{1}{q}\sqrt{\ell})$, with q the problem's number of optima. Individuals at Hamming distance smaller than 52.5 share their fitness.

When optimizing an Ising model, equally good partial solu-

tions should be kept in the population. Experiments show that an srGA with sharing and two-point crossover finds both optima quickly. For example, using population size 500, both optima are present in the population after about 79.78 generations. This result is obtained by taking the average over 50 independent runs. Adding a niching technique clearly helps an srGA running on an Ising model to avoid synchronization problems.

An obvious question is now: how does an srGA with sharing perform on easy symmetrical problems, like the TwoMax problem? A TwoMax benefits from genetic drift, and as soon as one niche disappears the algorithm converges quickly to an optimum. When sharing is used to keep both niches in the population, crossover between individuals from different niches produces poor offspring. This has a pernicious influence on the search process. For example, an srGA with sharing, population size 500, and two-point crossover is not able to find an optimum in 2000 generations. When using an undersized population that is not able to keep both niches in the population, sharing only prolongs the first phase of the search process. As soon as one niche disappears, the algorithm converges quickly to an optimal population containing only one optimum. In both cases sharing has no beneficial influence on the search process.

In contrary to an Ising model, a TwoMax becomes much harder when traditional niching techniques are used. The recombination operator becomes too disruptive. This is confirmed by the analysis of an srGA without niching running on a TwoMax. As soon as one niche disappears out of the population, the population's mean fitness increases drastically and the solution is found quickly. Adding a restricted mating technique that uses the Hamming distance as a measure to determine with which individual one can mate, to an srGA with sharing does not improve the performance of the recombination operator. The Hamming distance is not a good measure to separate the individuals with more ones from those with more zeros, and more complex search techniques are necessary. Good results for a TwoMax as well as for an Ising model are found in [11] where a clustering technique is used in combination with an univariate marginal distribution algorithm. The disruptive effect of recombination is eliminated by only allowing recombination between individuals of the same cluster and niching is induced by allowing each cluster to produce an amount of offspring proportional to their size or to their fitness average. In this way, the individuals from niche 0 and niche 1 have their own cluster, and since the individuals only mate within their own cluster, each cluster is actually solving a OneMax (or a ZeroMax).

6 SUMMARY AND CONCLUSION

This paper shows that there exists a key dividing line between two types of symmetrical problems: problems for which the GA benefits from the fact that genetic drift chooses between equally good partial solutions, and problems for which all equally good partial solutions have to be preserved to find an optimum. By analyzing the search process of a selectorecombinative GA optimizing a TwoMax and comparing this search process with that of an Ising model, we tried to understand the difference between these two types of symmetrical problems. For the first type of problem, searching in terms of superior building blocks is sufficient to find an optimum. For the second type of problems, searching in terms of non-inferior building blocks is necessary to find an optimum and can be obtained by using a niching technique.

In the context of designing a robust problem solver which solves problems of bounded difficulty, designing an algorithm which solves both types of symmetrical problems quickly, accurately, and reliably is not obvious. Naively adding fitness sharing as a standard component of a GA makes easy symmetrical problems like a TwoMax hard to solve. When using traditional niching techniques, the recombination operator is too disruptive, and more complex methods are necessary. Next to looking for more complex methods that solve both types of problems, we suggest looking for an a priori or runtime detector which discriminates between the two types of problems and selects the appropriate algorithm automatically.

Acknowledgments

The authors would like to thank Alessio Ceroni, Martin Pelikan and Kumara Sastry for many useful discussions. The first author is supported by the Flemish Institute for the Encouragement of Scientific and Technological Research in Industry – (IWT) (Flanders) (Belgium). The third author is a Post Doctoral Fellow of the Fund for Scientific Research – (FWO) (Flanders) (Belgium). The work was partly sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. Support was also provided by a grant from the U. S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0003. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, the U.S. Army, or the U.S. Government.

References

- [1] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 42–50. Morgan Kaufmann, 1989.
- [2] L. J. Eshelman and J. D. Schaffer. Crossover's niche. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 9–14. Morgan Kaufmann, 1993.
- [3] D. E. Goldberg. Four keys to understanding building-block difficulty, June 15 1998. Presented in Project FRACTALES Seminar at I.N.R.I.A. Rocquencourt, Le Chesnay, Cedex.
- [4] D. E. Goldberg. The design of innovation: Lessons from and for competent genetic algorithms. In press, 2002.
- [5] D. E. Goldberg and K. Deb. A comparative study of selection schemes used in genetic algorithms. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.
- [6] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multi-modal function optimization. In J. J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates, 1987.
- [7] J. H. Holland. Building blocks, cohort genetic algorithms, and hyperplane-defined functions. *Evolutionary Computation*, 8(4):372–391, 2000.
- [8] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm: I. continuous parameter optimization. *Evolutionary Computation*, 1(1):25–49, 1993.
- [9] B. Naudts and J. Naudts. The effect of spin-flip symmetry on the performance of the simple GA. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th Conference on Parallel Problem Solving from Nature*, volume 1498 of *LNCS*, pages 67–76. Berlin Heidelberg New York, 1998. Springer-Verlag.
- [10] C. K. Oei, D. E. Goldberg, and S. J. Chang. Tournament selection, niching, and the preservation of diversity. IlliGAL Report 91011, University of Illinois at Urbana-Champaign, 1991.
- [11] M. Pelikan and D. E. Goldberg. Genetic algorithms, clustering and the breaking of symmetry. IlliGAL Report 2000013, University of Illinois at Urbana-Champaign, 2000.
- [12] M. Pelikan and D. E. Goldberg. Hierarchical problem solving and the bayesian optimization algorithm. In D. Whitley, D. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, and H. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference 2000*, pages 267–274. Morgan Kaufmann, 2000.
- [13] D. Thierens. *Analysis and Design of Genetic Algorithms*. PhD thesis, Catholic University Leuven, Belgium, 1995.
- [14] C. Van Hoyweghen, D. E. Goldberg, and B. Naudts. Building block superiority, multimodality and synchronization problems. In Lee Spector et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference 2001*, pages 694–701. Morgan Kaufmann, 2001.
- [15] C. Van Hoyweghen, B. Naudts, and D. Goldberg. Spin-flip symmetry and synchronization. Submitted to the journal of Evolutionary Computation, 2001.
- [16] R. Watson, G. S. Hornby, and J. B. Pollack. Modeling building block interdependency. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th Conference on Parallel Problem Solving from Nature*, pages 97–106. Springer-Verlag, 1998.